Crocker

Network Working Group
Requests for Comments: 9

HOST SOFTWARE

G. Deloche, U.C.L.A.

1 May 1969

TABLE OF CONTENTS

1. Introduction

2. HOST-HOST Protocol
    2.1 Logical Links
        2.1.1 Primary
        2.1.2 Auxiliary
    2.2 Link Establishment
        2.2.1 General Procedures
        2.2.2 Example

3. Network Service Calls
    3.1 List of Service Calls

4. Data Structure
    4.1 "HOST" Table
    4.2 "LINK" Table
    4.3 "USER" Table

5. Network Program

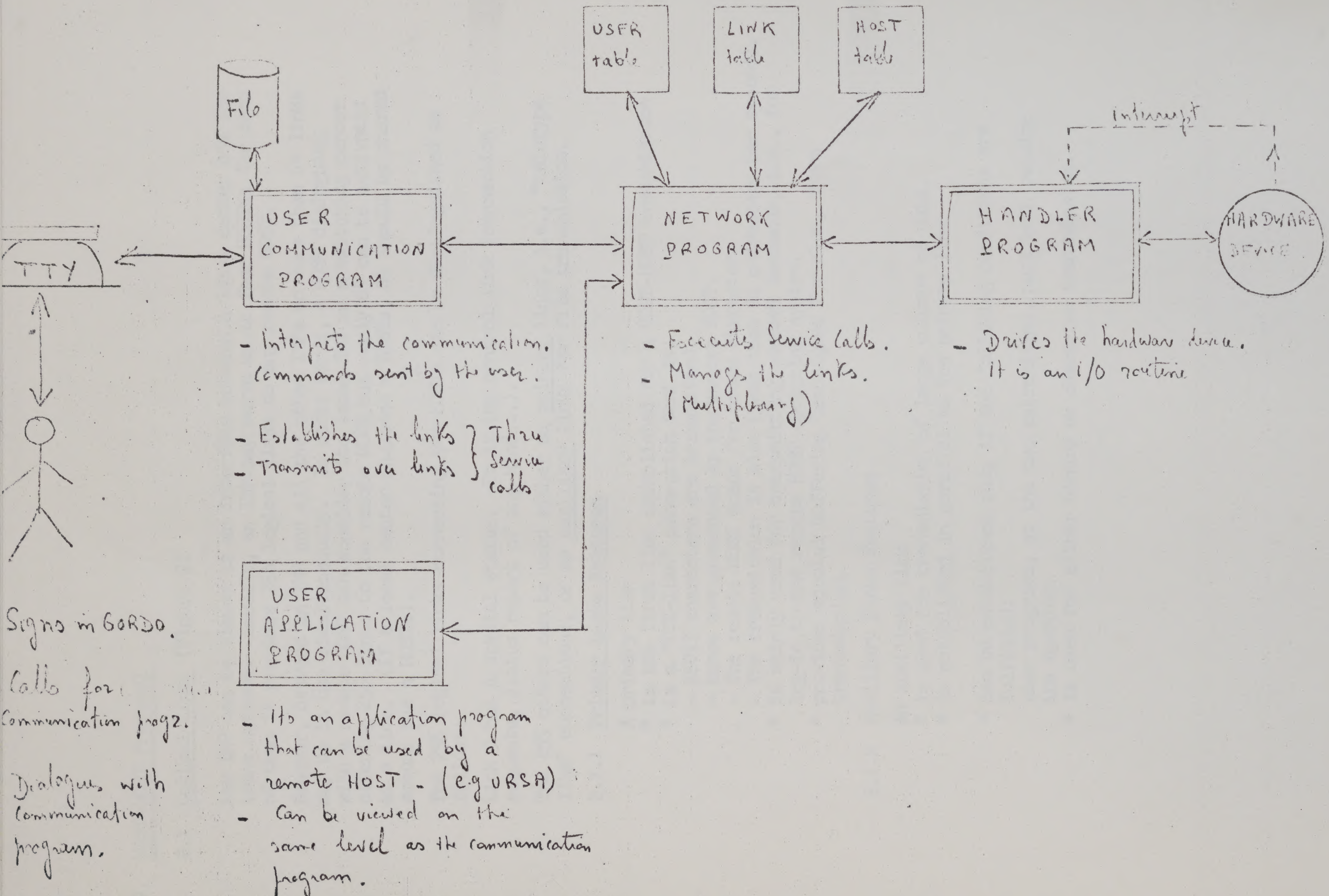
1. Introduction

This paper concentrates upon the HOST-HOST dialogue procedure.

Chapter 2 describes the logical links connecting the HOST, and the way data are exchanged over these links.

The emphasis of Chapters 3, 4, and 5 is on software organization and data structure.

Figure 1 highlights the different programs involved in a HOST.


USER table
LINK table
HOST table
File
interrupt
TTY
USER COMMUNICATION PROGRAM
NETWORK PROGRAM
HANDLER PROGRAM
HARDWARE DEVICE
- Interprets the communication. commands sent by the user.
- Establishes the links
- Transmits over links
Thru Service calls
- Executes Service Calls.
- Manages the links. (Multiplexing)
- Drives the hardware device. It is an i/o routine
USER APPLICATION PROGRAM
Signs in GORDO.
Calls for Communication progr.
Dialogues with Communication program.
- Its an application program that can be used by a remote HOST. (e.g URSA)
- Can be viewed on the same level as the communication program.

(Fig 1)

2. HOST-HOST Protocol

2.1 Logical Links (Figure 2)

Any IMP can be viewed as an interface between a local center and the trunk network. Locally, an IMP may serve up to four HOSTs; for each of them it provides 256 logical links to any remote HOST.

However, between an IMP and all the other IMPs no more than 64 links may be in use simultaneously. In other words, a HOST dialoguing with a remote HOST can consider its local IMP as a switching center offering 256 lines to the remote HOST, but only 64 can be activated at a time. (If a local center includes η HOSTs, 64 should be shared amount the η HOSTs).

The 256 logical links connecting two HOSTs can be distinguished as follows:

Link 0 has a special status. It is the control link (connection requests, status report of any kind...).

The 255 others can be used either as primary links, i.e., "teletype like" connections, or as auxiliary links for file transmission.

2.1.1 Primary Links Features

A primary link
* is the first link established for a HOST-HOST transmission.
* is a "TTY-like" connection that is:
  - ASCII characters are transmitted.
  - Echos are generated by the remote HOST.
  - The remote HOST scans for break character.
  - The transmission is slow (less than 20 characters per second).
* is mainly used for transmitting control commands, i.e., for log-in to the remote HOST operating system.
* provides special buffering techniques for slow, short transmission.

2.1.2 Auxiliary Links Features

An auxiliary link
* is used for transmission of large volumes of data.
* is established in parallel to the primary link

* can be established only if the following conditions are fulfilled:
  user programs, at the two extremities, must both require its opening.
* is used for either binary or character transmission.

2.2 Link Establishment

2.2.1 General Procedures

Each HOST(X) user will respect the following procedure for communicating with HOST(Y).

(a) Establish a primary link to HOST(Y).
A primary link is established to HOST(Y) through the control link 0. The connection is then in a pre-log-in state, i.e., the remote HOST expects its standard log-in procedures.

(b) Log-in Sequence
Standard ASCII characters are sent/received over the primary link. In that way, the HOST(X) user signs in to remote HOST(Y) by using its standard log-in procedures.

(c) Establish an auxiliary link to HOST(Y)
This establishment must be executed by both extremities. As in (a), this is done by using the control link 0.

(d) Send/Receive Text over Auxiliary link

2.2.2 Example

Figure 3 focuses on the data exchanged over the links during a HOST(X)-HOST(Y) dialogue.
HOST(X) has the network identification 8.
HOST(Y) has the network identification 5.

Notations Used:
* Circled stuffs represent characters, e.g. ENQ
* Parenthesised numbers are used for cross referencing with further explanations, e.g. (2)

Explanations

* (1) and (2) constitute the primary link establishment
-HOST(X) sends the following message over link 0:
" ENQ PRIM 0 1 2 OPT "

ENQ : Enquiry for link establishment (ASCII character)
PRIM : Link type: primary (Special Character)
0 1 2 : Logical link identification number in decimal (3 ASCII characters)
OPT : Options: it is an alphanumerical character, e.g. 9 . Possible options could be: Full Echo, data type...

-HOST(Y) acknowledges by sending back:
" ACK ENQ PRIM 0 1 2 OPT "

ACK : postive acknowledgement (ASCII character)-Link 12 is now established.
ENQ PRIM 0 1 2 OPT : The previous message is returned to the requestor for security purpose.

* (3) and (4) constitute a trivial example of a log-in procedure. -See remark 2 below-

*(5): HOST(X), talking to the operating system of HOST(Y), requests for URSA. URSA is supposed to be a user application program in HOST(Y).

*(6) and (7) constitute the auxiliary link establishment. After (5) an auxiliary link should be established. This is done by HOST(X) since it has the higher identification number in network. e.g., 8 against 5.
The procedure is very much like (1) and (2)

*(8): HOST(X) transmits a "file" to URSA. The transmission is done over link 25 which has just been established.

*(9): HOST(Y) answers back with a "file" over link 25.
And the dialogue goes on...

*(10): HOST(X) frees the links he has established
EOT : End of transmission (ASCII character).
⓪⓪②: Number of links wanted to be closed (3 ASCII character)
⓪①②⓪②⑤: Link identification number (ASCII characters)

*(11) HOST(Y) acknowledges back as in (2), (7).

Remark 1: The figure 3 doesn't show the heading of each message which are of course transmitted over these links. The characters represented on each line should be viewed inserted in the text zone of a message.

Remark 2: These characters -see (3) and (4)- can either be transmitted one at a time over the line (each character constitutes the text of a message) or be packed before transmission by the user communication program.

In either case, the remote HOST can consider the link as a normal teletype (Searchs breaking characters, provides echos...).

Remark 3: In (2), (7), or (11), HOST(Y) can answer back a negative acknowledgement character NAK instead of ACK. This, for many various reasons such as bad transmission, HOST(X) wants to open a link already established, and so forth. The message could be NAK IND where IND is a character indicating why the previous block has been refused. Upon receiving back such negative acknowledgements, HOST(X) will repeat its message until HOST(Y) accepts it. An emergency procedure will take place if too many successive NAK occur.

* (3) and (4) constitutes a typical example of a logon procedure -See remark 2 below-

*(5): USER(1), calling in the operating system of HOST(2), requests FTP USER1, which is supposed to be a user selectable program in HOST(2).

*(6) and (7) constitute the ancillary link establishment. After (5) an ancillary link should be established. This is done by HOST(2) since it has the higher identification number in network, e.g., 8 against 5.
The procedure is very much like (1) and (2).

*(8): HOST(2) transmits a "file" to USER. The transmission is done over link 23 which has just been established.

*(9): USER(1) answers back with a "file" over link 25. And the dialogue goes on...

*(10): HOST(2) frees the links it has established
(20): End of transmission (ASCII character).
(20/2): Number of links wanted to be closed (2 ASCII characters)
(23,25/2): Link identification number (ASCII characters)

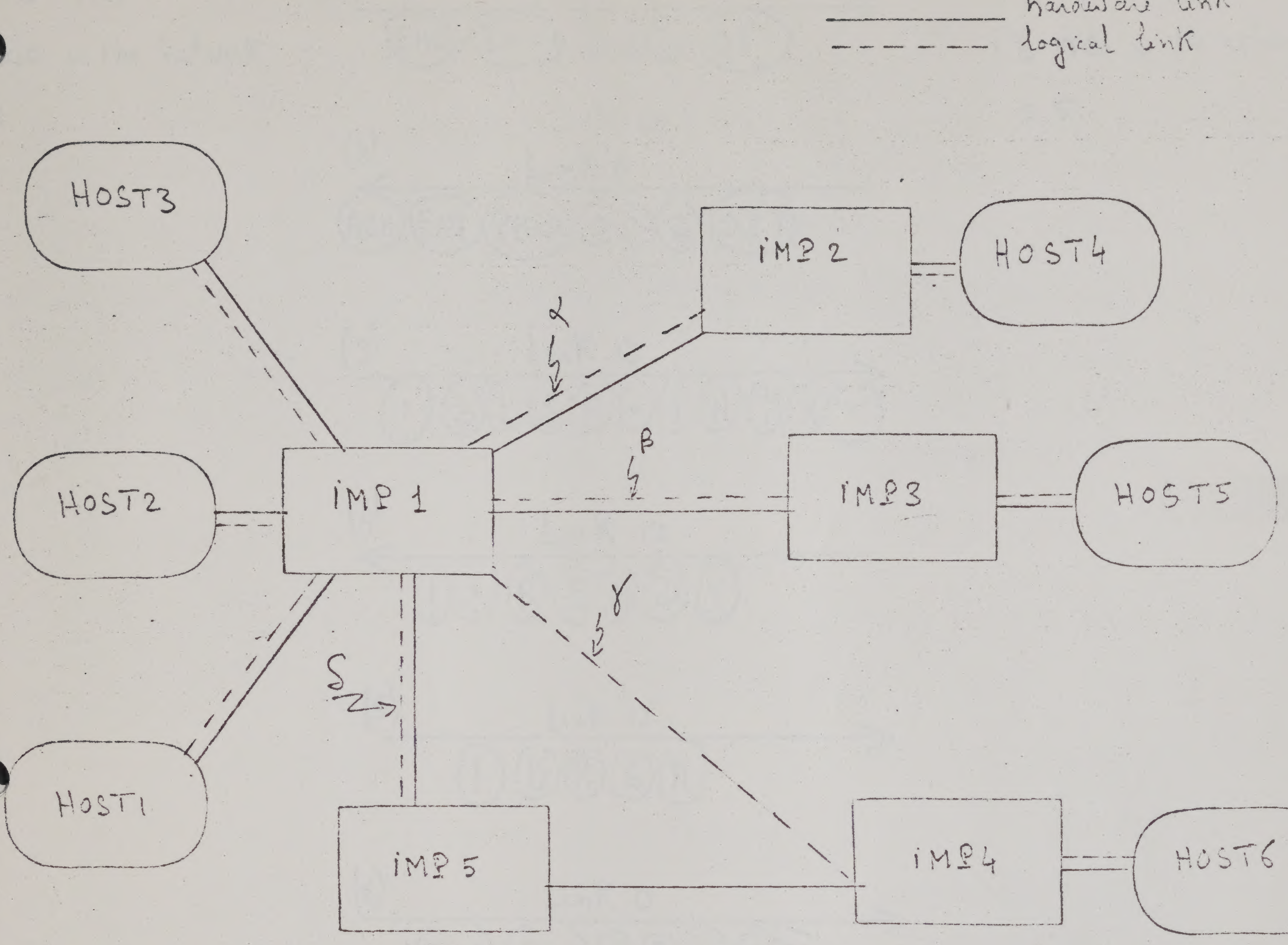
*(11) USER(1) acknowledges back as in (2), (7).

Remark 1: The figure 3 doesn't show the heading of each message which are of course transmitted over these lines. The characters represented on each line should be viewed appended to the text part of a message.

Remark 2: These exchanges -and (3) and (4)- can either be transmitted one at a time over the line (each "exchange" constitutes the text of a message) or packed before transmission by the user communication program.

In either case, the remote HOST can consider the line as a normal teletype (carriage returning characters, possible echos,...).

Remark 3: In (2), (7), or (11), USER(1) can answer with a negative acknowledgment character (NAK instead of ACK). Thus, for many various reasons such as non trivial error, HOST(2) wants to have a link already established, and so forth, the message would be (NAK) (nn) where (nn) is a character indicating why the previous logon has been refused. Upon receiving back such negative acknowledgment, HOST(2) will repeat its message until it's correct i.e. An error procedure will take place if necessary according the error.


hardware link
logical link
HOST3
HOST2
HOST1
IMP 1
IMP 2
HOST4
IMP3
HOST5
IMP 5
IMP4
HOST6
α
β
γ
δ

256 logical links between HOST1 and HOST5
256 logical links between HOST1 and HOST6
256 logical links between HOST1 and HOST4
" " " " HOST1 and HOST2

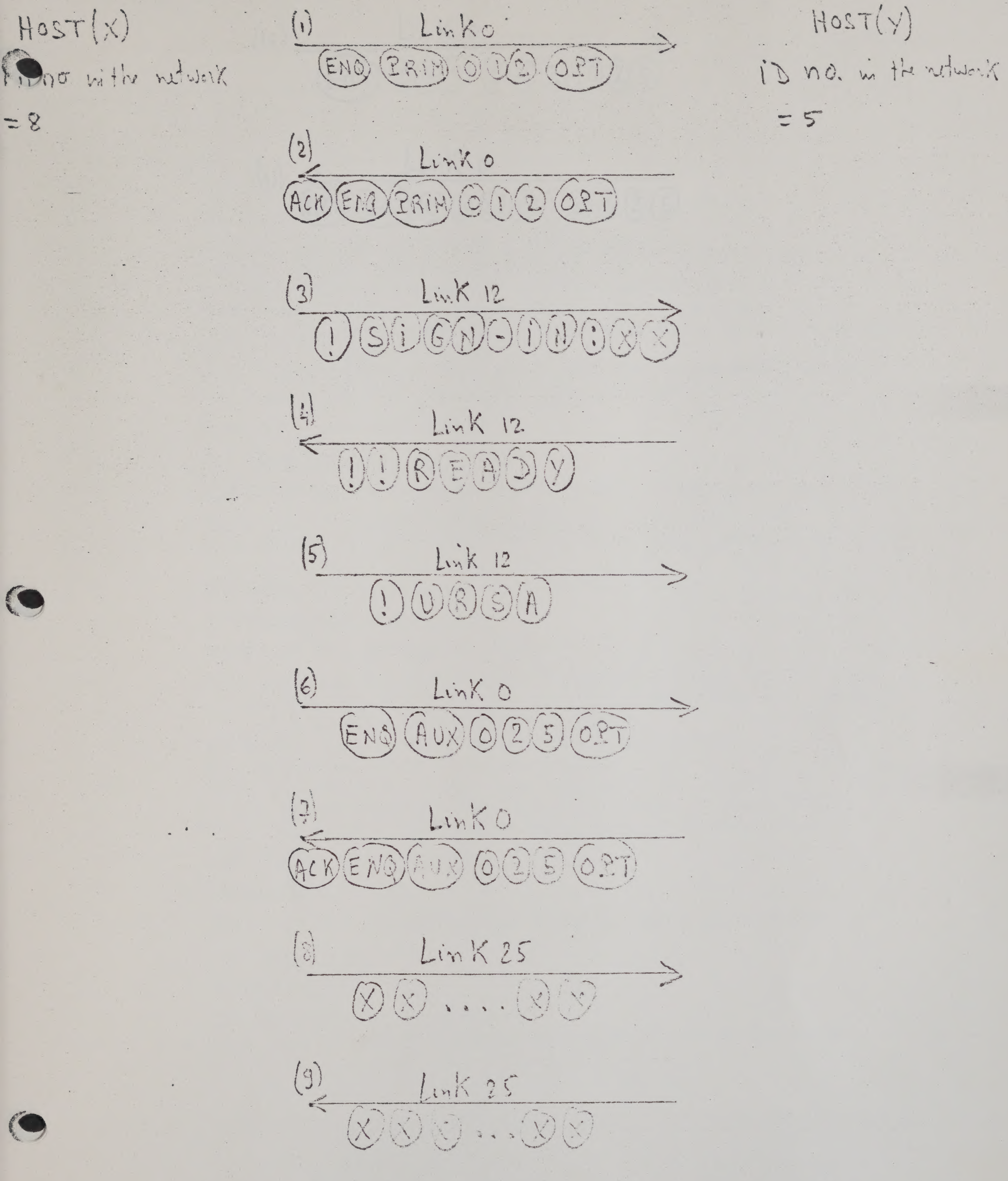
.. . .

256 logical links between HOST2 and HOST4
" " " " HOST2 and HOST6

. . . .

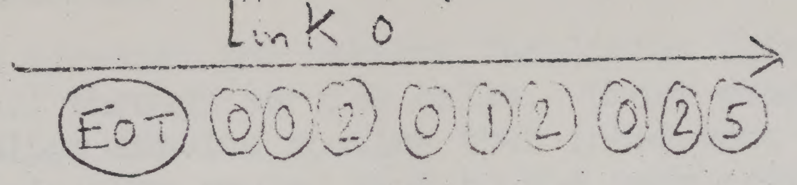
but $\alpha + \beta + \gamma + \delta = 64$ at a given time

(Fig 2)

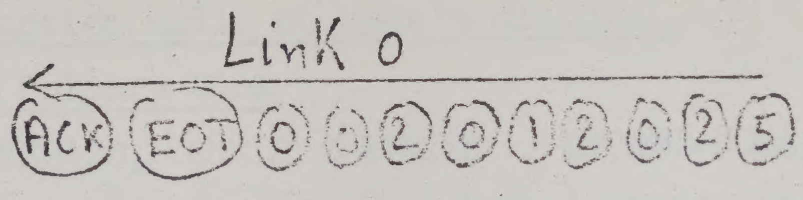

HOST(X)
ID no within network
= 8
HOST(Y)
ID no. in the network
= 5
(1)
Link 0
ENQ PRIM 0 1 2 OPT
(2)
Link 0
ACK ENQ PRIM 0 1 2 OPT
(3)
Link 12
! S i G N - I N : X X
(4)
Link 12
! ! R E A D Y
(5)
Link 12
! U R S A
(6)
Link 0
ENQ AUX 0 2 5 OPT
(7)
Link 0
ACK ENQ AUX 0 2 5 OPT
(8)
Link 25
X X . . . . X X
(9)
Link 25
X X X . . . X X

(Fig 3)

(10)


Link 0
EOT
0
0
2
0
1
2
0
2
5

(11)


Link 0
ACK
EOT
0
0
2
0
1
2
0
2
5

(Fig 3)

3. Network Service Calls

A user program accesses the network facilities (link establishment, data transmission...) through service calls. Under execution, a service call traps to a monitor service routine that interprets and executes the service. Control is then routed back to the user program.

3.1 List of service calls at user's disposal.

(a) Open Primary Link

OPENPRIM(PRIMID,HOSTID, BUFFADDR , INTRPT-CODE,[OPT])

PRIMID: User identification of the primary link.
HOSTID: Remote HOST identification.
BUFFADDR: Buffer address for the incoming messages.
INTRPT-CODE: Code that the network program should give to the user program when he is interrupted because a message has come back.
OPT: Options such as "full echo" (for testing purpose), message required after successful link establishment, etc....

Remark: [ ]: not required.

(b) Open auxiliary link

OPENAUX(AUXID,PRIMID,BUFFADDR,INTRPT-CODE,[OPT])

AUXID: User identification of the auxiliary link.
PRIMID: User identification of a primary link. Refers to an already established primary link.
BUFFADDR, INTRPT-CODE, OPT same meaning as above.

(c) Transmission over link

TRANSLINK (ID, BUFFADDR, N, [OPT])

ID: User link identification. Depending on which type of links we want to transmit, this identification number will be equal to a previously defined AUXID/PRIMID.
BUFFADDR: Data location address for transmission.
N: Data bytes number for transmission.
OPT: Options such as data type (character vs. binary), acknowledgements required (utilization of the auxiliary links in a half duplex mode), trace bit, etc....

(d) Modify link parameters

MODIFLINK (ID, OPT)

ID: User link identification (Equal to either AUXID/PRIMID)

(e) Close link

CLOSE LINK (ID, [OPT])

ID: Same meaning as above.
OPT: Can be used to close all the links in use by the user.

4. Data Structure

The allocation and the management of the links are carried out by means of three tables:

A Table Sorted By HOST.
A Table Sorted By LINK.
A Table Sorted by USER.

4.1 HOST Table (See Figure 4)

It is a bit-table indicating, for a given remote HOST, which links are free. (bit-0 means free link)

This table should provide 256 bits per HOST (256 logical links possible). At a given time no more than 64 bits can be set to 1 in the whole table.

4.2 LINK Table (See Figures 4 and 5)

This table contains as many sections as links in use.Figure 5 describes the structure of a section.
Starting and retrieval are carried out dynamically upon using a hashing technique based on the network link identifications.

4.3 USER Table (See Figure 4)

The table structure is given on Figure 4. These are as many sections as active users. Each section contains the user identification (given by the operating system) and the identifications of the links in use by this user. Notice that a link has two identifications: that of the user (given as a parameter in the OPEN service call) and that of the network (that is attributed by the network program).

This table is hashed by users.

5. Network Program

The emission functions of the network programs are fulfilled by monitor service routines. In that sense, this program can be viewed as belonging to the operating system.

These functions are concerned with the link establishments and data transmission; they are started by the service calls previously described.

Let's explain how these routines allocate and manage the links by describing the operations involved during the execution of the OPENPRIM routine.

Suppose that the value of the parameter HOSTID is equal to j.

(a) j is used as an index for the "HOST" table to reach the "HOST j"

(b) In "HOST j" section, we select the first free link (First bit=0) e.g., i^{th} bit.

(c) j and i determine respectively the HOST-IMP destination and the network link number.

(d) This j-i value is used as a hashing code to open a new section in the link table. e.g. section ℓ.

(e) In this section ℓ, the link ID zone is filled up with j-i, the "link opened by us" and "primary" bits are set to 1. (See Figure 5.)

(Remark: It is only when we receive back the acknowledgement message from the remote HOST - See Figure 3: (2)-that the link is considered completely established. Then we set to 1 the bit "link established".). Also in this section ℓ, we store the parameter BUFFADD Value in the "buffer address zone", and the user identification number, implicitly given, in "the user ID zone".

(f) Using the user identification number, we hash the USER Table to open (or find) the right m section.

We update this m section by storing the user link ID number (PRIMID) and the network link ID number (i).

(g) We prepare the message text:
(ENQ) (PRIM) (0) (0) (i) (OPT)

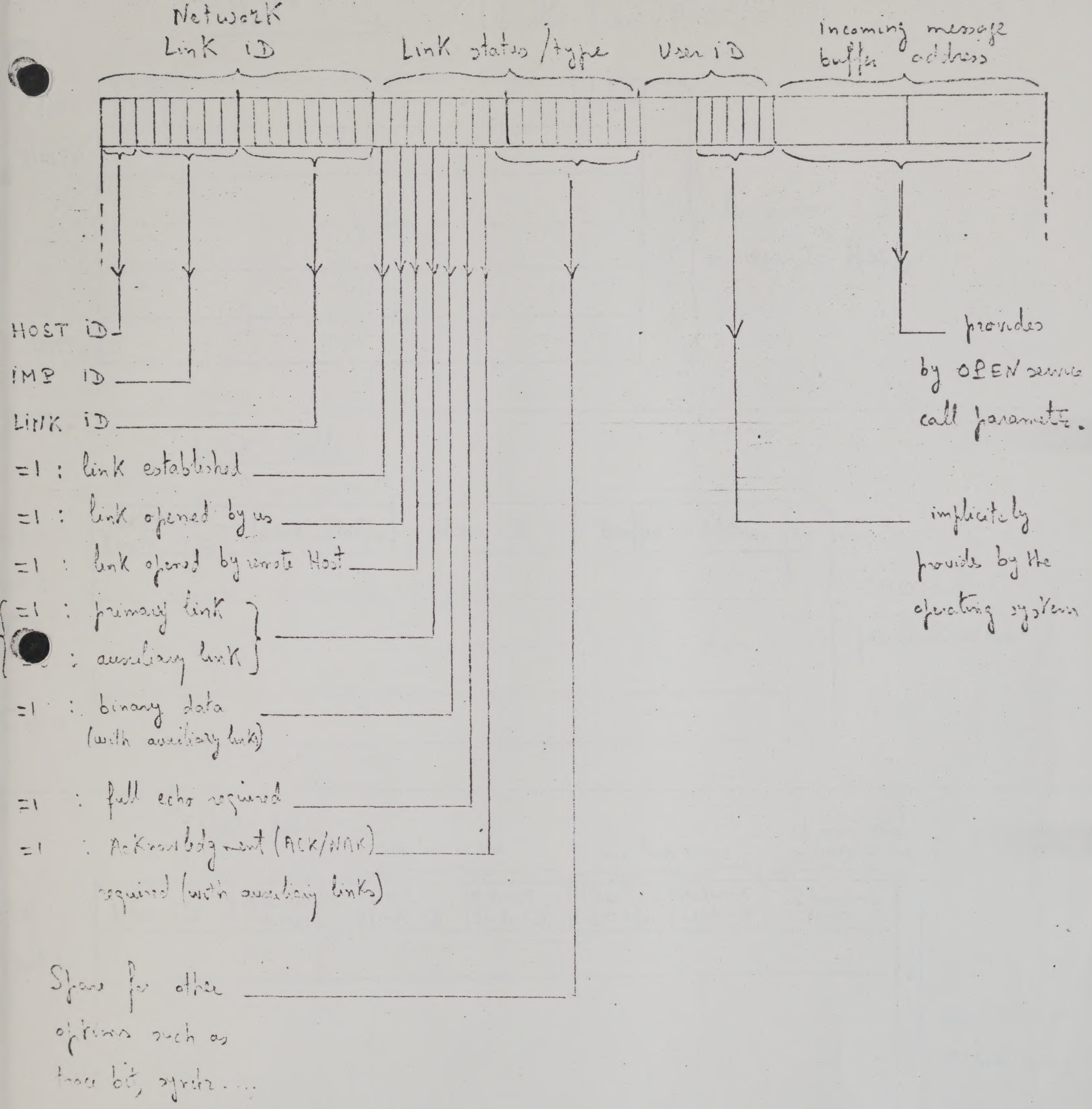
(h) We prepare a heading according to BBN specifications (in order to send the message over link 0).

(i) We calculate the HOST checksum.

(j) We put together the heading, checksum, text by providing marking.

(k) We queue up this message for the handler.

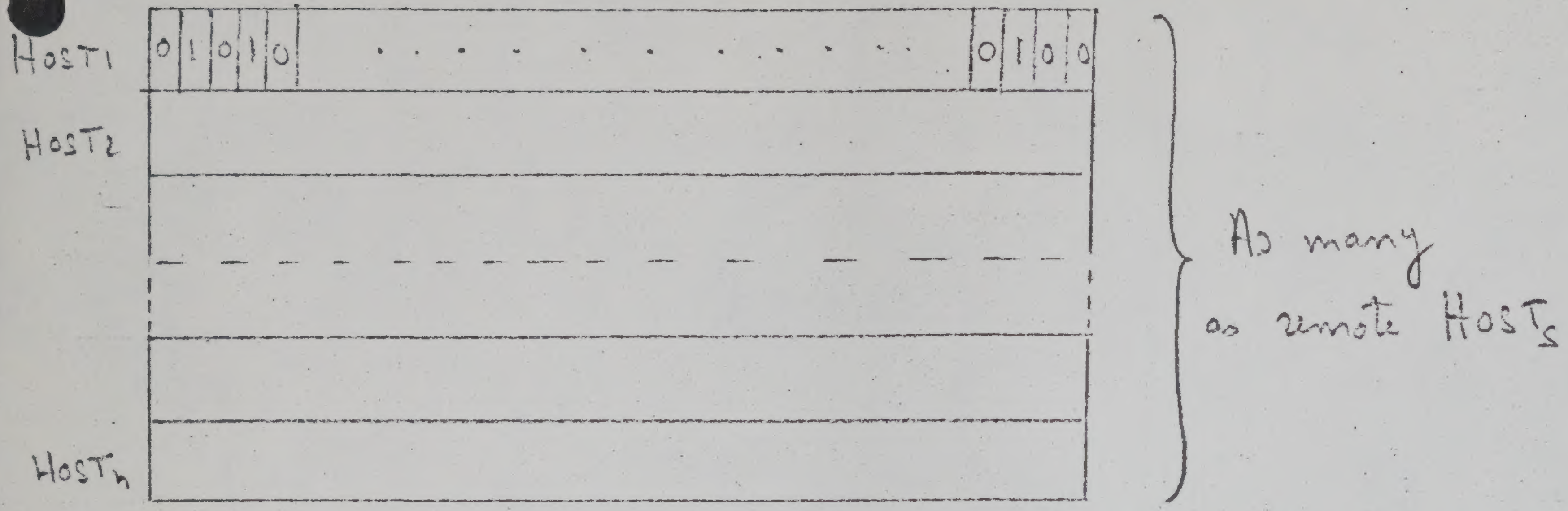
The receiving functions will use these tables in a very similar way.


Network Link ID
Link status/type
User ID
Incoming message buffer address
HOST ID
IMP ID
LINK ID
=1 : link established
=1 : link opened by us
=1 : link opened by remote Host
=1 : primary link
: auxiliary link
=1 : binary data (with auxiliary links)
=1 : full echo required
=1 : Acknowledgment (ACK/NAK) required (with auxiliary links)
Space for other options such as trace bit, syntax ....
provides by OPEN service call parameter.
implicitely provids by the operating system

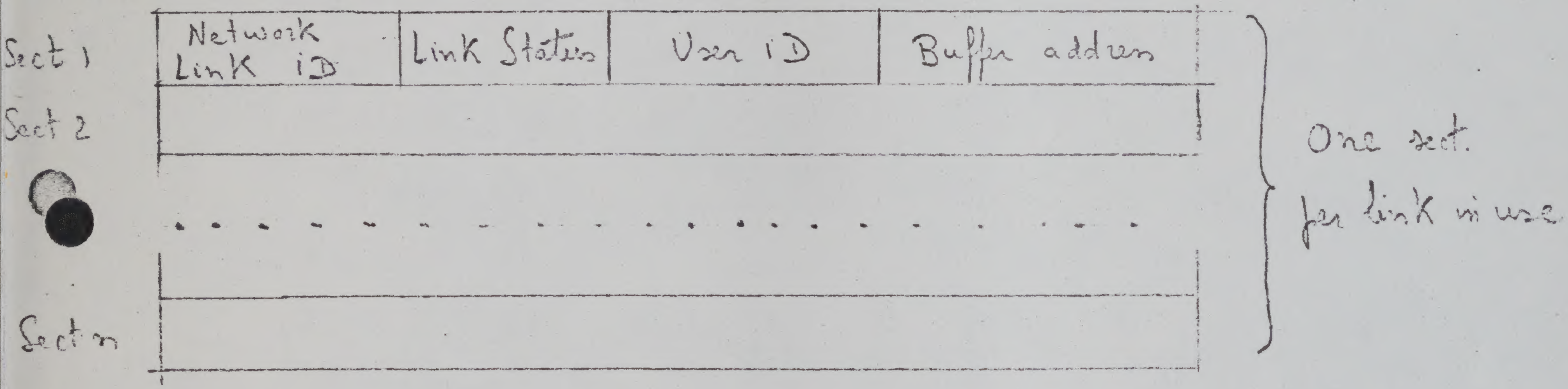
Link table structure

(Fig 5)

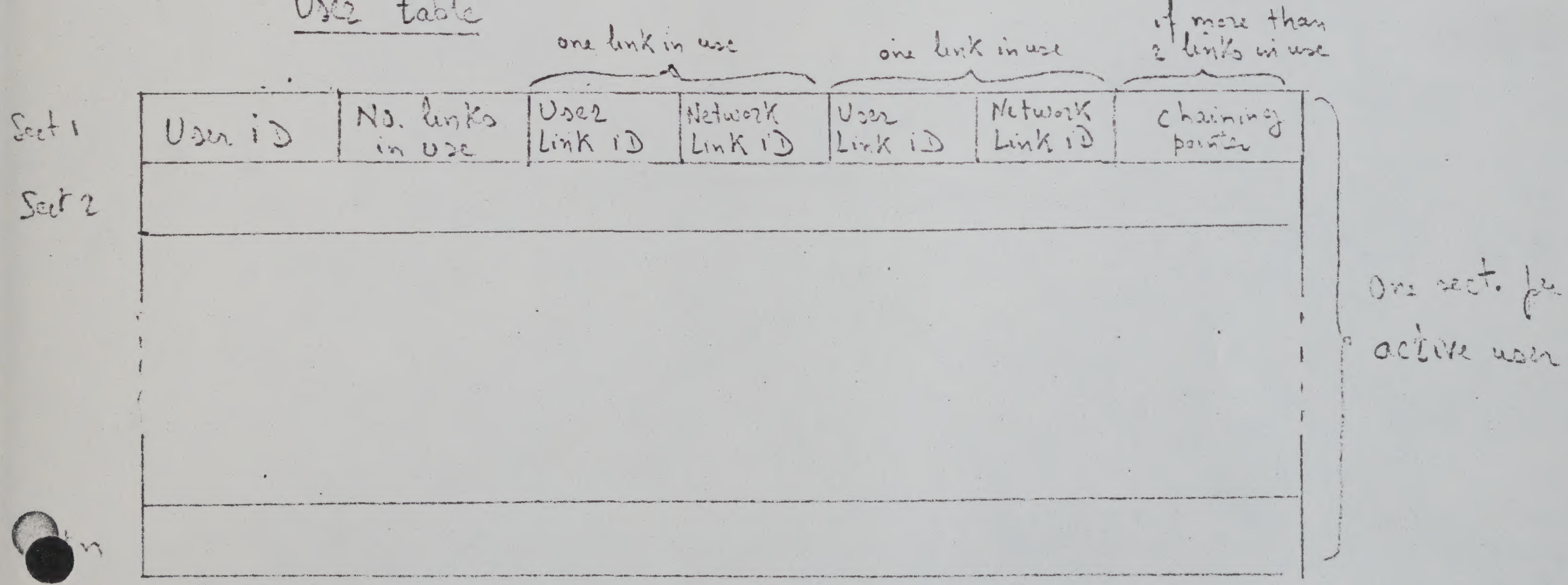
"Host" table (Link-free/HOST)


HOST1
0 1 0 1 0
0 1 0 0
HOST2
HOSTn
As many as remote HOSTs

"Link" table


Sect 1
Network Link ID
Link Status
User ID
Buffer address
Sect 2
Sect n
One sect. per link in use

"User" table


one link in use
one link in use
if more than 2 links in use
Sect 1
User ID
No. links in use
User Link ID
Network Link ID
User Link ID
Network Link ID
Chaining pointer
Sect 2
n
One sect. per active user

(Fig 4)